JoindIn

https://joind.in/talk/64d74

The full story of Software Bill of Materials

Lars Bendix, Lund University
Andreas Göransson, QCM
sneSCM.org

Agenda

- DevOps SBoM motivation
- SBoM history

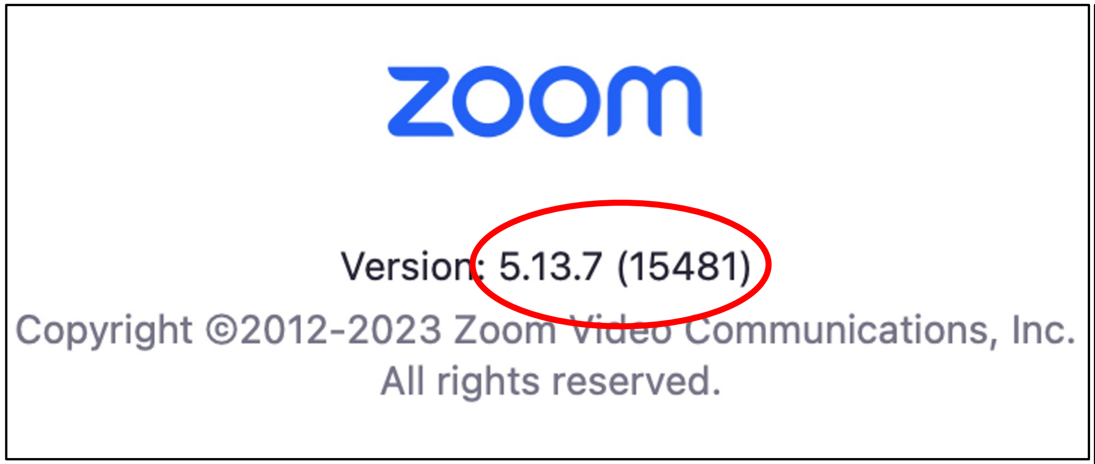
- SBoM - Use Case categories

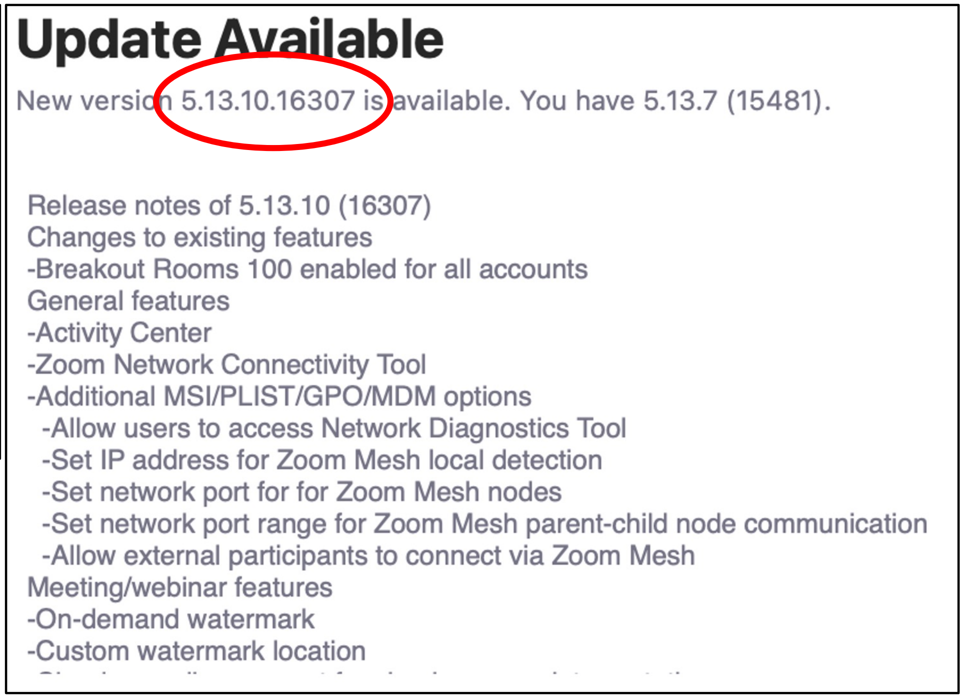
- General SBoM considerations
- Lessons learned

Motivation

- Will be an external requirement - US & EU
- Should have been an internal requirement since the 1980’s
- Dev are the optimal producers of SBoMs
- Ops and Dev are heavy users of SBoMs
- SBoMs needed for all systems in environment (know what you have)
- If you know what you have, then change control (and cyber security) is easier
- An SBoM tells you what to roll back to
- How you can use/exploit SBoM - the full story

These are examples of SBoMs

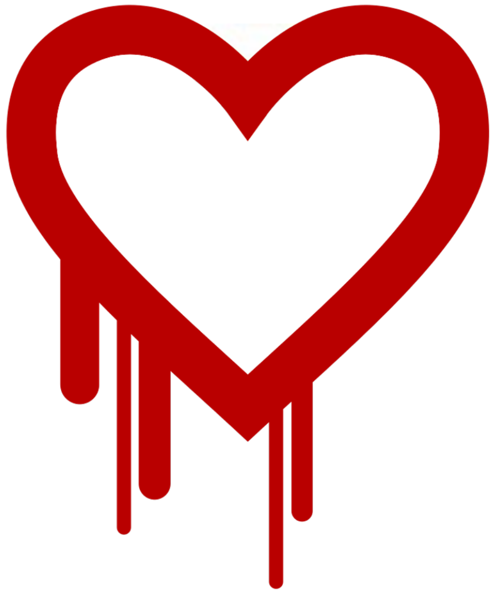

Version: 5.13.7 (15481)
Copyright ©2012-2023 Zoom Video Communications, Inc.
All rights reserved.


Update Available

New version 5.13.10.16307 is available. You have 5.13.7 (15481).

Release notes of 5.13.10 (16307)
Changes to existing features
-Breakout Rooms 100 enabled for all accounts
General features
-Activity Center
-Zoom Network Connectivity Tool
-Additional MSI/PLIST/GPO/MDM options
  -Allow users to access Network Diagnostics Tool
  -Set IP address for Zoom Mesh local detection
  -Set network port for for Zoom Mesh nodes
  -Set network port range for Zoom Mesh parent-child node communication
  -Allow external participants to connect via Zoom Mesh
Meeting/webinar features
-On-demand watermark
-Custom watermark location

SBoM anno 2020s



- Executive Order 14028 signed in 2021
- Strengthen the US' ability to respond quickly and efficiently to cybersecurity vulnerabilities
  - Heartbleed, Solarwind, Colonial Pipeline hack
- US Department of Commerce and NTIA
  - The Minimum Elements For a Software Bill of Materials
  - "List of ingredients" with focus on vulnerability scans

Fast rewind - Back to the future

SBoM anno 1980s - part I

Wayne Babich: “Many times the fastest approach to finding a bug is not analysis of the program itself, but analysis of the *history* of the program – how it was created. The history of the program is called its *derivation*.”

A precise *derivation* of a program or module requires:

- an identification of the *tool* that created it
- an identification of the data that was *input* to the tool
- an identification of the *options* and *arguments* given to the tool
- the *reason* why that particular data, arguments, and options were given to the tool
- the *person* who was responsible for creating the data
- the *date* and *time*

An ounce of derivation is worth a pound of analysis!

Back to the future II

MICHAEL J. FOX
CHRISTOPHER LLOYD
Getting back was only the beginning.
STEVEN SPIELBERG Presents
BACK TO THE FUTURE PART II
A ROBERT ZEMECKIS Film
MICHAEL J. FOX
CHRISTOPHER LLOYD BACK TO THE FUTURE PART II LEA THOMPSON
THOMAS F. WILSON MUSIC BY ALAN SILVESTRI PRODUCTION DESIGN BY RICK CARTER
DIRECTOR OF PHOTOGRAPHY DEAN CUNDEY, A.S.C. EXECUTIVE PRODUCERS STEVEN SPIELBERG FRANK MARSHALL
KATHLEEN KENNEDY SCREENPLAY BY BOB GALE STORY BY ROBERT ZEMECKIS & BOB GALE
PRODUCED BY BOB GALE AND NEIL CANTON DIRECTED BY ROBERT ZEMECKIS
A UNIVERSAL PICTURE
OPENS NOVEMBER 22ND

Imaged by Heritage Auctions, HA.com

SBoM anno 1980s - part II

- Clearmake automatically create SBoMs called Configuration Records
- Re-use build artifacts like object files in a smart and secure way
- Configuration Record contains information about input files, build environment and output files
- Configuration records can be read and used by machines and audited manually by developers

Overview - Types SBoM categories

- Bo-Materials
  - Search for object by UID
- Bo-Process
  - Reuse
  - Debugging
  - Rebuild
  - Build audit
- Bo-Information
  - Licence tracking
  - Export control
  - Legal aspects
  - Test-related matters
  - Information sharing

BoM use case category

Search for an object by UID

Software Composition Analysis (SCA):

- Are features 12, 15 and 19 included in this binary for QA?
- We get a new (binary) patch - is it already in there?
- What is “operating” on our systems (Ops)

Logon module - source code ver. 2.1 (compiled on several occasions)

SBoM = **BoM**

BoP use case category

Reproducibility - or producibility - is a core concept for configuration management

For that we need the exact source code - obviously…

But that is not sufficient:

- Escrow development
- Fixing a bug in a 13-year old petro-chemical installation

To the degree of what you consider “identical” we must include the process:

- Tools, versions, options, environments, …

SBoM = BoM + **BoP**

BoI use case category

Information about the artifacts in the SBoM needed for communication, audits, certifications, …

Test related matters:

- In case a program or system can affect human safety, test information can be vital to keep in an SBoM
  - Test cases, test results, test environments (HW and SW), …
- Possible need to provide proof that test cases have been performed in a legal dispute
- Test information can also be used as a “quality stamp”
  - Our software passes these test and therefore complies to regulation X

SBoM = BoM + BoP + **BoI**

Specific attack example - thanks to Giulio Vian

An internal agent replaces the source code or a tool that is used for producing the build with something malicious.

We must have UIDs not just for binary, but also source code and tools - and…

Can we trust our (internal) UIDs?

SBoMs can facilitate "build paranoia" - allows two different entities to use the SBoM to create the binary and compare the result.

SBoM may not be 100% bullet proof cyber security, but they can be part of the solution.

https://reproducible-builds.org

SBoM considerations - part I

- Detail of SBoM:
    - what data is needed?
    - washing/hiding/filtering info that you are not allowed to reveal
    - representation format
- Availability of SBoM
    - “Document” (man and machine readable)
    - “Query” (internal/external)
- Automation - both in creation and using
- Static and dynamic data

SBoM considerations - part II

- Keeping the SBoM up to date:
    - when I patch Firefox or Windows
    - build- , instal- and run-time parameterization/configuration
- SBoM for tools and environments too - and Everything
- SBoM and the Cloud
    - version of compiler?
    - SBoM for cloud services (and microservices)?

Key takeaways

- DevOps as **consumers**:
  - List of ingredients & Vulnerability scan
  - 100s of use cases, 10+ categories, 3 types
- DevOps as **producers**:
  - “Builders” of a binary are at the origin of (most) SBoM data
  - Automation is at the heart of DevOps
- Doesn’t it look a little like:
  - Configuration Items (attributes and relations)
  - Configuration Status Accounting

SBoM = BoM + BoP + BoI

Q&A

https://fileadmin.cs.lth.se/cs/Personal/Lars_Bendix/Research/SBoM/

https://joind.in/talk/64d74